


UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA CHAMPAIGN
BOOKSTACKS



Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

330
B385
1991:125 COPY 2

STX

Input Control in Job Shops

The Library of the
MAY 16 1991
University of Illinois
of Urbana-Champaign

N. Raman
Department of Business Administration

BEBR

FACULTY WORKING PAPER NO. 91-0125

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

March 1991

Input Control in Job Shops

N. Raman

Department of Business Administration
University of Illinois at Urbana-Champaign
Champaign, IL 61820

ABSTRACT

Input Control plays a critical role in regulating the release of jobs into a production system. For a just-in-time system, we model the input control decision as a bicriterion problem which requires the maximization of the sum of job release times subject to minimum total job tardiness.

The proposed solution method uses a constructive approach in which jobs are grouped into blocks for simultaneous processing. The membership of each block is updated in an iterative manner by solving a sequence of subproblems. Our computational experience indicates that this approach yields substantial improvement in job release times without affecting the tardiness values adversely. This result is of interest to an operating manager who is responsible for controlling work-in-process levels and earliness costs in addition to meeting due dates effectively.

1 Introduction

In many manufacturing systems, the release of jobs into the production shop floor is regulated by means of an input control mechanism. The purpose of this mechanism is to allow jobs to be introduced selectively into the system instead of permitting all of them in as and when they become available. In most materials requirements planning (MRP) systems, this mechanism results in important savings. First, the work-in-process levels are reduced leading to less congestion and confusion on the shop floor. Second, by deferring jobs required at a future date, the system is less susceptible to demand variations, and is therefore, less nervous. Third, it simplifies scheduling and smooths the output obtained from the system.

Input control is also imperative in many flexible manufacturing systems (FMS) because of the limited buffer space available. Input control may also be an integral part of the hierarchical planning and scheduling of an FMS. Other instances when input control is required are for shop load leveling (Irastorza and Deanne 1974), load balancing (Shimoyashiro et al. 1984), part and tool loading in an FMS (Stecke 1983), etc.

While there are many practical benefits of using input control, Baker (1984a) notes some of the possible drawbacks as well. By removing some of the available options to the scheduler, input control may be counterproductive. Specifically, from a scheduling perspective, if the scheduling objectives relate to *regular* measures, i.e., if the schedule costs are nondecreasing in job completion times, then delaying job input could easily result in suboptimal solutions.

The just-in-time (JIT) approach, however, promotes the notion that jobs should be released as close to their due dates as possible even if they are available for scheduling much earlier. Many scheduling objectives relevant to JIT systems, such as minimizing earliness, are nonregular. Given a set of jobs available for scheduling at any point in time, the basic approach adopted in such systems is to delay the release of *noncritical* jobs as much as possible while insuring at the same time that *critical* jobs are taken up without delay. Therefore, given a set of jobs available for scheduling, the problem facing the production planner is to identify the critical jobs which need to be input immediately. In addition, (s)he needs to determine the future release times for the deferred jobs.

In this paper, we model this decision as the problem of maximizing the release times

of nonurgent jobs subject to the minimization of the sum of tardiness of urgent jobs. This bicriterion problem considers two objectives ordered hierarchically. The primary objective is to minimize the sum of tardiness of urgent jobs. Subject to this requirement, the secondary objective deals with the maximization of the release times of nonurgent jobs. Note that while the primary objective is a regular measure of performance, the secondary objective is nonregular.

While this is the first known attempt to combine these two objectives, the literature on the job shop tardiness problem considered independently is quite rich. In view of the exponential complexity of this problem (see, for example, Rinnooy Kan 1976), prior research on this problem has primarily considered the use of priority dispatching rules. [See Baker 1984b and Vepsalainen and Morton 1987 for an extensive discussion on this research.] In a recent study, Raman, Talbot and Rachamadugu (1989) developed an implicit enumeration approach as well as a decomposition based heuristic for solving this problem.

While the problem of maximizing the sum of job release times in a job shop has not been addressed directly, there is considerable literature on this subject for single machine systems. Note that for a single machine system, this objective reduces to that of minimizing earliness. This problem is shown to be NP-complete by Chand and Schneeberger (1986). Baker and Scudder (1990) provide a survey of the work done on this objective and its variants. Ahmadi and Bagchi (1987) have extended this objective to the case of a two-machine flow shop. This paper is organized as follows. In §2, we present an integer programming formulation of the input control problem and discuss its decomposition. The solution procedure is discussed in §3. Our computational experience with this approach is described in §4. We conclude in §5 with a discussion of the experimental results. The notation used in this paper is given in Appendix 1.

2 The Input Control Problem

The input control problem can be formulated as below.

ICP

$$\text{Maximize } z_2 = \sum_{j=1}^N r_j \quad (1)$$

subject to

$$\sum_{j=1}^N T_j = z_1, \text{ where } z_1 = \min_{\sigma \in \mathcal{S}} \{T(\sigma)\} \quad (2)$$

$$x_{uv} - x_{ji} + M(1 - \delta_{jiuv}) - p_{uv} \geq 0, \quad \forall j, u \in \mathcal{J}, \text{ and } (j, i), (u, v) \in \mu_m, \quad m = 1, \dots, M \quad (3)$$

$$x_{ji} - x_{uv} + M\delta_{jiuv} - p_{ji} \geq 0, \quad \forall j, u \in \mathcal{J}, \text{ and } (j, i), (u, v) \in \mu_m, \quad m = 1, \dots, M \quad (4)$$

$$x_{jl} - p_{jl} \geq \begin{cases} r_j & \text{if } l = 1 \\ x_{ji} & \text{otherwise; } \forall j, i, \text{ and } (i, l) \in \mathcal{O}_j \end{cases} \quad (5)$$

$$x_{jN_j} + E_j - T_j = d_j, \quad \forall j \quad (6)$$

$$\delta_{jiuv} \in \{0, 1\}, \quad \forall j, i, u, v; \quad x_{ji} \geq 0 \quad \forall j, i; \quad r_j, E_j, T_j \geq 0, \quad \forall j \quad (7)$$

where x_{ji} is the completion time of operation i in job j , and δ_{jiuv} equals 1 if operation i in job j precedes operation v in job u , and is zero, otherwise.

In this formulation, Equation (1) refers to the secondary objective of maximizing the sum of job release times. Constraint (2) specifies the primary objective of minimizing total tardiness. In the given expression, $T(\sigma)$ refers to the tardiness incurred in schedule σ , and \mathcal{S} refers to the set of all schedules. Constraints (3) and (4) are the disjunctive constraints that ensure that no more than one operation is processed on a machine at any time. Constraints (5) ensure that any operation can be scheduled only after its predecessor operation is completed, and the first operation of any job is started only after the job is released. Constraints (6) define tardiness while Constraints (7) specifies the nature of the variables.

ICP generalizes two well-known problems. The primary objective addresses the minimization of mean tardiness in a job shop. If it is feasible to complete all jobs by their due dates, then the problem reduces to maximizing the sum of job release times subject to all jobs being completed by their due dates. Maximizing $\sum_{j \in \mathcal{J}} r_j$ is equivalent to minimizing $\sum_{j \in \mathcal{J}} (d_j - r_j)$. Also,

$$(d_j - r_j) = \sum_{i=2}^{N_j} (x_{ji} - x_{j,i-1}) + p_{j1}.$$

If we denote the waiting time of job j after operation i by w_{ji} , then

$$x_{ji} - x_{j,i-1} = w_{j,i-1} + p_{ji}.$$

Hence,

$$(d_j - r_j) = (d_j - c_j) + \sum_{i=1}^{N_j-1} w_{ji} + \sum_{i=1}^{N_j} p_{ji}$$

The last term on the right hand side is a constant. Hence, minimizing $\sum_{j \in \mathcal{J}} (d_j - r_j)$ is equivalent to minimizing $\sum_{j \in \mathcal{J}} (d_j - c_j) + \sum_{j \in \mathcal{J}} \sum_{i=1}^{N_j-1} w_{ji}$. This objective has been studied by Ahmadi and Bagchi (1987) for a two machine flow shop. In the case of a single machine job shop, this objective reduces to minimizing total earliness which is a well researched problem (see, for example, Baker and Scudder 1990).

Both job shop mean tardiness problem (see, for example, Rinnooy Kan 1976) and the single machine earliness problem (Chand and Schneeberger 1986) have been individually shown to be NP-complete. Consequently, we need to use heuristic methods for solving any problem of reasonable size. The proposed approach is based on identifying a sequence of subproblems within ICP, and constructing a schedule by solving these subproblems iteratively.

3 Embedded Subproblems

In the rest of the paper, we will assume that all jobs are numbered in the order of increasing due dates, i. e., if $v > u$, then $d_v \geq d_u$. Intuitively, it is clear that jobs that are either already tardy or are in the imminent danger of being so need to be released into the system immediately, while the input of nonurgent jobs can be deferred. Consider such a partition of \mathcal{J} into the set of *urgent* jobs \mathcal{J}_1 and the set of *nonurgent* jobs \mathcal{J}_2 , $\mathcal{J}_1 \cup \mathcal{J}_2 = \mathcal{J}$, and $\mathcal{J}_1 \cap \mathcal{J}_2 = \emptyset$; and a schedule $\sigma = (\mathcal{J}_1, \mathcal{J}_2)$ in which all jobs in \mathcal{J}_1 are completed before any job in \mathcal{J}_2 is started, i. e.,

$$\max_{j \in \mathcal{J}_1} \{c_j\} < \min_{j \in \mathcal{J}_2} \{r_j\}$$

Let $c(\cdot)$ denote the completion time of jobs in a given set in schedule σ . Then,

Remark 1 *In an optimal solution, if $\sum_{j \in \mathcal{J}} T_j > 0$, then*

$$i) \sum_{j \in \mathcal{J}} T_j = \sum_{j \in \mathcal{J}_1} T_j,$$

$$ii) T_j < c(\mathcal{J}_1 \setminus j) + p_j - d_j, \forall j \in \mathcal{J}_1, \text{ and}$$

$$iii) \mathcal{J}_1 \text{ starts at time zero.}$$

PROOF: Refer to the Appendix 2.

For a given partition $(\mathcal{J}_1, \mathcal{J}_2)$ that satisfies the conditions given in Remark 1, ICP separates into two problems, ICP1 that considers only \mathcal{J}_1 and ICP2 that considers only \mathcal{J}_2 as given below.

ICP1(\mathcal{J}_1)

$$\text{Maximize } z_2 = \sum_{j \in \mathcal{J}_1} r_j \quad (8)$$

subject to

$$\sum_{j \in \mathcal{J}_1} T_j = z_1 \quad (9)$$

and

$$(3) - (7), \forall j \in \mathcal{J}_1.$$

ICP2(\mathcal{J}_2)

$$\text{Maximize } z_2 = \sum_{j \in \mathcal{J}_2} r_j \quad (10)$$

subject to

$$T_j = 0, \forall j \in \mathcal{J}_2 \quad (11)$$

$$r_j - \max_{j \in \mathcal{J}_1} \{c_j\} \geq 0 \quad \forall j \in \mathcal{J}_2 \quad (12)$$

and,

$$(3) - (7), \forall j \in \mathcal{J}_2.$$

Constraints (11) and (12) insure that the partition $(\mathcal{J}_1, \mathcal{J}_2)$ is feasible. Note that the dual objectives apply only to ICP1, while ICP2 considers only the objective of maximizing the sum of job release times.

In general, any schedule σ optimal to ICP will have inserted idle times because the secondary objective is a non-regular measure of performance. This results in job *blocks* as shown in Figure 1. Two jobs j_1 and j_n belong to the same block if σ consists of a subsequence (j_1, j_2, \dots, j_n) such that

$$c_{j_l} \geq c_{j_{l-1}} > r_{j_l}, \quad l = 2, \dots, n$$

It follows that for two adjacent blocks, $B_{[k]}$ and $B_{[k+1]}$,

$$\max_{j \in B_{[k]}} \{c_j\} \leq \min_{j \in B_{[k+1]}} \{r_j\}.$$

Note that if \mathcal{J}_1 is not empty, then it constitutes a block.

ICP2 can now be reformulated in terms of blocks as follows:

$$\text{Maximize } z_2(ICP2) = \sum_{k=1}^{J_2} \sum_{j \in B_{[k]}} r_j \quad (13)$$

subject to

$$\sum_{k=1}^{J_2} y_{jk} = 1, \quad \forall j \in \mathcal{J}_2 \quad (14)$$

$$\max_{j \in B_{[k]}} \{c_j\} - \min_{j \in B_{[k+1]}} \{r_j\} \leq 0, \quad \forall k \quad (15)$$

$$\min_{j \in B_{[1]}} \{r_j\} - \max_{j \in \mathcal{J}_1} \{c_j\}, \quad (16)$$

$$x_{uv} - x_{ji} + M(1 - \delta_{jiuv}) - p_{uv} \geq 0, \quad \forall j, u \in B_k, \text{ and } (j, i), (u, v) \in \mu_m, \quad m = 1, \dots, M \quad (17)$$

$$x_{ji} - x_{uv} + M\delta_{jiuv} - p_{ji} \geq 0, \quad \forall j, u \in B_k, \text{ and } (j, i), (u, v) \in \mu_m, \quad m = 1, \dots, M \quad (18)$$

$$x_{jl} - p_{jl} \geq \begin{cases} r_j & \text{if } l = 1 \\ x_{ji} & \text{otherwise; } \forall j, i, \text{ and } (i, l) \in \mathcal{O}_j \end{cases} \quad (19)$$

$$x_{jN_j} + E_j - T_j = d_j, \quad \forall j \in B_k, \forall k \quad (20)$$

$$\delta_{jiuv} \in \{0, 1\}, \quad \forall j, i, u, v; \quad x_{ji} \geq 0, \quad \forall j, i; \quad r_j, E_j, T_j \geq 0, \quad \forall j \quad (21)$$

For a given partition of \mathcal{J}_2 into K blocks, $\{B_k\}_{k=1}^K$, that satisfies (14)–(16), ICP2 decomposes into K independent subproblems; each subproblem addresses, independently for each block, the maximization of $\sum_{j \in B_k} r_j$ subject to (17)–(21). We shall, henceforth, refer to this problem as the *block maximum release time problem* (BMRP). For a given sequence σ_k of jobs in block B_k , we define the block release time R_k , block completion time C_k , block due date D_k and block processing time P_k as:

$$R_k = \min_{j \in B_k} \{r_j\}$$

$$C_k = \max_{j \in B_k} \{c_j\}$$

$$D_k = \min_{j \in B_k} \{d_j | c_j = C_k\}$$

$$P_k = \max_{j \in B_k} \{c_j\} - \min_{j \in B_k} \{s_j\}$$

Clearly, for the partition and the schedule of jobs in each block to be feasible, constraints (14)–(16) need to be satisfied, where (15) can now be restated as:

$$C_{[k]} - R_{[k+1]} \leq 0, \quad \forall k$$

In addition, the following results can be established for an optimal schedule.

Remark 2 *In each block B_k , $k = 1, \dots, K$ in an optimal schedule, there exists one critical job j such that*

$$c_j = d_j.$$

PROOF: Refer to the Appendix 2.

We will denote the completion time and due date of the critical job in block B_k by $C_{k\bullet}$ and $D_{k\bullet}$, respectively.

4 Solution Approach

A heuristic solution procedure can be constructed utilizing the results obtained in the previous section in the following manner. First, we assume that all jobs can be finished on time, and set $\mathcal{J}_2 = \mathcal{J}; \mathcal{J}_1 = \emptyset$. Next, a set of initial blocks is constructed through a maximal partition of \mathcal{J}_2 , and the maximum release time problem is solved for each block. If there is no feasible solution for a given block B_k (which implies that at least one job in B_k cannot be completed by its due date in this schedule), all jobs in B_k are transferred to set \mathcal{J}_1 , and \mathcal{J}_2 is updated accordingly. At the end of this step, a tentative partition of \mathcal{J} into \mathcal{J}_1 and \mathcal{J}_2 , and a partition of \mathcal{J}_2 into blocks that, considered *independently*, consist of early jobs is obtained.

Next, problem ICP1 is solved for jobs in \mathcal{J}_1 . Subsequently, we solve a relaxation of ICP2 obtained by ignoring (15) and (16) with respect to the blocks in \mathcal{J}_2 by considering them in the increasing order of their indexes, and sequencing them such that $C_{k\bullet} = D_{k\bullet}$ for each k . If this sequence satisfies (15) and (16), then from Remark 4, it is optimal to ICP2, and the algorithm terminates. Otherwise, if (15) is violated by (say) block $B_{k+1}, k \geq 1$, then it is merged with block B_k . This requires solving BMRP for B_k with the enlarged set of jobs,

and updating the values of $C_{k\bullet}$ and $D_{k\bullet}$. Block B_k is sequenced such that $C_{k\bullet} = D_{k\bullet}$. This exercise is repeated for all such blocks that violate (15).

On the other hand, if (16) is violated, then jobs in B_1 are transferred to \mathcal{J}_1 , and ICP1 is re-solved with the respect to the new set of jobs in \mathcal{J}_1 . The indexes of blocks in \mathcal{J}_2 is updated accordingly, and the process is repeated until (16) is satisfied with the revised makespan of \mathcal{J}_1 .

This procedure will terminate within $|\mathcal{J}| - 1$ steps yielding a sequence which provides a feasible partition of \mathcal{J} into \mathcal{J}_1 and \mathcal{J}_2 , and also insures that the blocks in \mathcal{J}_2 satisfy the conditions stated in Remark 4. A formal statement of the algorithm is given in Appendix 3. The major steps in the algorithm are now discussed.

4.1 Determination of Initial Blocks

The initial set of blocks is obtained from a *maximal* partition of \mathcal{J} . A partition is called maximal if

1. each block in the partition satisfies the following condition: If u and v are, respectively, the smallest and the largest job indexes in the block, then $d_v - p_v < d_u$, and
2. $\min_{j \in B_k} \{d_j - p_j\} \geq \max_{j \in B_{k-1}} \{d_j\}$, $k = 2, \dots, K$

It can be shown that in an optimal solution, if the job that is completed the last in each block has the largest due date among all jobs in that block, then the resulting partition is maximal. An $O(N^2)$ algorithm for obtaining such a partition is given in Appendix 3.

4.2 The Block Maximum Release Time Problem

For each block, we next solve the block maximum release time problem (BMRP) – finding the sequence of jobs in each block that maximizes the sum of job release times subject to the requirement that they are completed by their due dates.

4.2.1 Initial Solution

The solution method proposed for BMRP is an improvement heuristic for which an initial solution is obtained by constructing a schedule for an equivalent problem – the completion time problem (CTP). The objective of CTP is to minimize the sum of completion times of jobs subject to ready times; the ready time of job j is given by $r'_j = d_{max} - d_j$, where d_{max} is the maximum of the due dates of jobs within the given block.

To see the equivalence between these two problems consider the problem shown in Figure 1. In this 3-job example, we have $d_3 < d_2 < d_1 = d_{max}$ for the BMRP. The corresponding instance of CTP is constructed by defining the following job ready times:

$$r'_1 = 0, r'_2 = d_1 - d_2, \text{ and } r'_3 = d_1 - d_3$$

INSERT FIGURE 1 ABOUT HERE

Denoting the variables in CTP with primes, it can be seen that,

$$c'_{max} = c_{max}, \text{ and } c'_j = d_{max} - r_j.$$

Therefore, maximizing $\sum_j r_j$ in BMRP is equivalent to minimizing $\sum_j c'_j$ in CTP. BMRP can then be solved by first solving CTP to determine job completion times c'_1, c'_2 , and c'_3 , and then using the above relationship to obtain the ready times r_j .

While CTP remains a hard problem even for a single machine system (Hariri and Potts 1983), a feasible solution to it can be constructed quite easily. We generate a *non-delay* schedule by considering only those jobs which are available at a given machine when the scheduling decision is to be made at that machine. Ties between competing jobs are broken in favor of the job with the shortest imminent operation time. The order of operations generated for CTP is reversed to obtain the desired sequence of jobs within the batch for BMRP. This sequence determines the start times s_{ji} of each operation i within each job j for the given block at the appropriate machine.

4.2.2 Schedule Improvement

During the improvement phase, we consider one machine at a time, in order, starting with the bottleneck machine. For each machine, an attempt is made to reschedule operations, and

the revised schedule is then used to update the parameters considered for the next machine. Thus, this phase is iterative; it terminates when no further improvement is possible. The problem to be solved at each machine involves maximizing the sum of release times of operations subject to their completion by the specified due dates. Because this is a single-machine problem, maximizing the sum of release times is equivalent to minimizing total earliness. [We will, henceforth, refer to this problem as the earliness problem subject to release times (ERP).]

Note that an operation i in job j can not be taken up before its ready time r_{ji} which is given by,

$$r_{ji} = \begin{cases} c_{j,i-1} & \text{if } i > 1 \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, because we consider only one machine at a time and keep the sequences at other machines unchanged, the due date of operation d_{ji} is determined by

$$d_{ji} = \begin{cases} s_{j,i+1} & \text{if } i < N_j \\ d_j & \text{otherwise} \end{cases}$$

Let I be the total number of operations on any given machine. For the ease of presentation, we introduce the following notation. Let a_i , b_i , t_i , f_i and e_i be, respectively, the ready time, due date, processing time, completion time and earliness of operation i . ERP can now be stated as:

$$\text{Minimize } v = \sum_{i=1}^I e_i$$

subject to,

$$f_i - t_i \geq a_i, \quad \forall i$$

$$e_i = b_i - f_i$$

$$f_i - f_h + M(1 - \delta_{hi}) - t_i \geq 0, \quad \forall i, h, i \neq h$$

$$f_h - f_i + M(1 - \delta_{hi}) - t_i \geq 0, \quad \forall i, h, i \neq h$$

$$f_i, e_i \geq 0, \quad \forall i$$

where, δ_{hi} equals 1 if operation h precedes i , and is zero, otherwise.

Given the complexity of the regular earliness problem, it is unlikely that a polynomial time exact solution algorithm is available for ERP. We construct a heuristic solution algorithm which is a modification of the approximation method suggested by Chand and Schneeberger (1985) for the regular earliness problem. For the sake of completeness, we give an outline of the Chand and Schneeberger heuristic (CH) below.

The algorithm builds the schedule backwards by using the Smith heuristic, i. e., the iterative step involves selecting, from among all operations feasible for assignment, the operation with the smallest t_i . The schedule developed in this manner is optimal if

$$t_{[i]} \geq t_{[i+1]}, \quad i = 1, \dots, I - 1.$$

Suppose $\sigma^1 = \{i, \sigma\}$ and $\sigma^2 = \{h, i, \sigma\}$ are two subsequences that satisfy the above optimality condition. However, at the next step the operation selected is g that yields the subsequence $\sigma^3 = \{g, h, i, \sigma\}$ that violates this condition. The heuristic then backtracks to the partial solution σ^1 to consider two alternative completions. In the first sequence, g is assigned to the position just ahead of i and the rest of the schedule is developed by using the Smith method. In the second completion, g is assigned to the next to last position among the unscheduled jobs and, subject to this assignment, the best sequence obtained by using the Smith method is found. The better of these two completions is selected.

When all operations are available at time zero, checking if any operation is feasible for assignment at a given position requires only insuring that $f_i \leq b_i$. However, with non-zero ready times, the algorithm needs to additionally insure that an operation does not start before its ready time. More importantly from the computational perspective, the assignment of an operation to a given position should insure that a feasible completion is available for the remaining subproblem, i. e., there is at least one schedule in which

$$f_i - t_i \geq a_i$$

for each unscheduled operation i .

Determining whether such a completion exists for a given partial sequence is equivalent to solving the problem of minimizing the maximum lateness of the unscheduled operations subject to ready times ($I/1/r_i \geq 0/L_{max}$). In particular, a feasible completion exists if and

only if the optimal $L_{max} \leq 0$. Although $I/1/r_i \geq 0/L_{max}$ is NP-hard in the strong sense, effective algorithms for solving it have been proposed by McMahon and Florian (1975) and Carlier (1982) that are based on implicit enumeration. We modify Carlier's algorithm in order to improve its efficiency in the context of ERP. This is done by identifying conditions that lead to infeasible completions. Clearly a positive lower bound to the overall problem gives one such condition. Our computational experience revealed that preprocessing to insure that the following two criteria are satisfied also improved the overall solution time:

1. $t \geq a_i + t_i$, for all unscheduled i , and
2. $t \geq \sum_{i \text{ unscheduled}} t_i$

where t is the start time of the most recently scheduled operation.

Modifying the start times and the completion times of an operation of any job on a given machine results in revising the ready time for the succeeding operation and the due date for the preceding operation for the same job. Thus, in general, rescheduling a machine leads to updating the parameters for the next machine to be considered. Starting with the bottleneck machine, we reschedule each machine in the order of their workloads in each cycle. The cycles are repeated until no further improvement is observed. Although it is difficult to theoretically guarantee convergence of the objective function value, we found it to be so in our computational experience; indeed, the bulk of the improvement was obtained within the first three cycles.

4.2.3 The ICP1 Problem

As discussed earlier, ICP1 is bicriterion problem with the primary objective of minimizing mean tardiness of all jobs in \mathcal{J}_1 . Our solution method consists of solving the mean tardiness problem first. The job due dates are then revised, if necessary, based on the resulting schedule. Given these due dates, the maximum release time problem is next solved following the procedure discussed in the previous section. Note that such a decomposition of the two objectives can result in a suboptimal solution if there are alternative optimal solutions to the mean tardiness problem.

In view of the strong NP-completeness of the mean tardiness problem, we propose using the heuristic method developed by Raman, Talbot and Rachamadugu (1989) which is briefly described now. This solution procedure is a schedule improvement procedure for which the initial solution is generated by applying the Modified Operation Due Date (See, for example, Baker 1984b) rule with operation due dates (ODDs) set loosely at the maximum values that they can assume without delaying the corresponding jobs. Each machine is then considered in the order of their relative workload, and an attempt is made to revise the schedule of operations on that machine by modifying their operation due dates. Jobs processed on all machines are ranked in the nonincreasing order of their tardiness. For any operation in a given job with positive tardiness, first we determine the appropriate interval for searching for the ODD. For each possible ODD value in this interval, the entire system is rescheduled. The value which yields the minimum total tardiness is returned as the ODD for that operation. This step is repeated for all other operations of that job processed on the machine under consideration, for all other tardy jobs on that machine following their rank order, and for all machines in the system.

Given the minimum mean tardiness schedule, the revised job due dates are given by

$$d_j \leftarrow \max(d_j, c_j).$$

Subject to these due dates, we solve BMRP for the jobs in \mathcal{J}_1 . Because we already have a feasible schedule initially, we can go directly to the schedule improvement phase discussed in Section 4.2.2.

Note that solving the mean tardiness problem can result in zero tardiness. In such instances, it may be possible to defer the release time of jobs in \mathcal{J}_1 .

5 Computational Experience

Two sets of experiments were conducted. The first set evaluated the improvement obtained in the sum of job release times by solving the dual-objective input control problem over the single-objective mean tardiness problem. While it is clear that explicit consideration of this objective should result in better solutions, the purpose of this set of experiments was to examine the margin of possible improvement achieved by using the suggested algorithm.

The objective of the second set of experiments was to judge the effectiveness of the proposed solution method with respect to known upper bounds for the case in which all jobs can be completed on time. From the due date constraints, we have $r_j + p_j \leq d_j$, for $\forall j \in J_2$. Hence,

$$\sum_{j \in J_2} r_j \leq \sum_{j \in J_2} d_j - \sum_{j \in J_2} p_j = UB_0$$

However, UB_0 provides a weak upper bound. Ahmadi and Bagchi (1987) derive a stronger bound for the case of a two-machine flow shop in which all job due dates are equal. To our knowledge, this is the only other upper bound available currently for multiple machine problems addressing similar objectives. We will denote this upper bound by UB . The second set of experiments, consequently, addressed a 2-machine flow shop with equal job due dates and compared the solution value yielded by MSP with respect to UB .

5.1 Experimental Design

The first set of experiments considered a 5-machine system with 20 jobs. Each job was assigned 5 operations, and the machine visitation sequence was assigned randomly though successive operations of a given job were processed in different machines. Operation processing times varied uniformly in the interval (0,99). The due date of d_j of a given job j was determined by

$$d_j = F(\sum_j p_j)$$

F was sampled from a uniform distribution in the interval $(F - RF/2, F + RF/2)$. F and R respectively control the tightness and the variability of job due dates. In this study, two sets of due date tightness levels were used. Four levels of tight due dates were obtained by setting F at 0.05, 0.1, 0.15 and 0.20. These values resulted in positive tardiness values (although at the value of 0.20, the tardiness values were very small). Similarly, loose due dates at four levels were obtained at F values of 0.30, 0.40, 0.50, and 0.60. At each of these eight levels, two values of R —0.5, and 1.5, were used to provide a total of 16 combinations of due date tightness and variability.

Five instances of each problem scenario were randomly generated. Each instance was solved using two approaches. The first approach employed the solution procedure discussed

in Section 3 that considered both primary and secondary objectives. The second approach considered only the primary objective of minimizing total tardiness. The solution values with respect to total tardiness and the sum of job release times obtained under both approaches were recorded and averaged over the five problem instances for reporting purposes. In order to restrict the computational costs within reasonable limits, the Modified Operation Due Date (MOD) rule [see, for example, Baker (1984)] was used for solving the mean tardiness problem under both approaches. In total, the first set of experiments considered 80 problems.

The second set considered a 2-machine flow shop. The smaller problem considered 25 jobs while the larger problem had 50 jobs. The operation processing times were selected from a uniform distribution in the interval (0,99). All jobs had the same due date d which was determined by

$$d = F(\sum_j p_j).$$

Five values of F —0.6, 0.7, 0.8, 0.9, and 1.0 were used to generate increasingly loose due dates. [As reported in Ahmadi and Bagchi’s (1987) study as well, we found that $F < 0.6$ led to due date infeasibility in many cases.] As in the first set of experiments, five problem instances were solved for each scenario. For each instance, the ratio of the sum of job release times obtained by solving ICP to the upper bound derived by using Ahmadi and Bagchi’s approach was recorded. The results report the average as well as the minimum and the maximum values of these ratios over the five problem instances. In all, the second set of experiments considered 50 problems.

5.2 Experimental Results

Tables 1 and 2 give the results of the first experiment. These tables compare the performance of ICP and MOD with respect to total tardiness and the sum of job release times. For better comparison, the reported total tardiness values are normalized with respect to the sum of job processing times. Similarly, the sum of job release times is normalized with respect to the sum of job due dates. In both tables, z_1 , and z_2 denote normalized total tardiness and normalized sum of release times respectively. Table 1 gives the results for the case of tight due dates, while Table 2 deals with the loose due dates. In both tables, the last column

reports the average solution time for the proposed algorithm on an IBM 3090-600 mainframe computer.

Table 1 indicates that ICP results in substantial improvement in z_2 values over MOD even when due dates are quite tight. This improvement generally increases as due dates become progressively looser. Note also that ICP gives marginally better total tardiness values as well. This is due to two factors. First, because of the job indexing scheme, ICP employs a second tie-breaking rule. Whenever two or more operations are found to have the same MOD value at the time a scheduling decision is to be made, it selects the operation for the job with the earlier due date. Second, note that ICP first schedules jobs in \mathcal{J}_1 , while MOD considers all jobs to be schedulable at any given time. Therefore, in constructing a nondelay schedule, MOD frequently takes up a nonurgent job for processing (if by doing so, machine idleness is avoided). This may delay the processing of an urgent job that arrives at the machine soon thereafter.

It can be seen from Table 2 that relative performance of ICP over MOD is superior at higher values of R . This is due to the fact that, with greater dispersion in job due dates, the final schedule contains a larger number of blocks. Hence, a larger portion of jobs is completed close to their due dates with consequent increase in release times as well.

Table 3 presents the results of the second set of experiments. The values reported are the ratios of the ICP solution value to the upper bound UB . The results indicate the effectiveness of the algorithm for solving ICP2, at least in the case of a 2-machine flow shop. In all problem instances, the solution value is found to be within 4.8% of the upper bound.

6 SUMMARY

This paper examines the effectiveness of employing input control as a mechanism for deferring job release in a just-in-time system. The input control decision is modeled as a dual objective problem of minimizing total job tardiness and maximizing the sum of job release times in a lexicographic manner.

The proposed solution method uses a constructive approach in which jobs are grouped into blocks for simultaneous processing. The membership of each block is updated in an iter-

ative manner by solving a sequence of subproblems. Our computational experience indicates that this approach yields substantial improvement in job release times without affecting the tardiness values adversely. This result is of interest to an operating manager who is responsible for controlling work-in-process levels and earliness costs in addition to meeting due dates effectively.

Appendix 1

Notation

j	Job index, $j = 1, \dots, N$
m	Machine index, $m = 1, \dots, M$
d_j	Due date of job j , $j = 1, \dots, N$
p_j	Processing time of job j , $j = 1, \dots, N$
r_j	Release time of job j , $j = 1, \dots, N$
s_j	Start time of job j , $j = 1, \dots, N$
c_j	Completion time of job j , $j = 1, \dots, N$
\mathcal{O}_j	Set of pairs of adjacent operations in job j , $j = 1, \dots, N$
N_j	Number of operations in job j , $j = 1, \dots, N$
T_j	Tardiness of job $j = \max(0, c_j - d_j)$, $j = 1, \dots, N$
E_j	Earliness of job $j = \max(0, d_j - c_j)$, $j = 1, \dots, N$
p_{ji}	Processing time of operation i in job j , $j = 1, \dots, N$, $i = 1, \dots, N_j$
r_{ji}	Release time of operation i in job j , $j = 1, \dots, N$, $i = 1, \dots, N_j$
c_{ji}	Completion time of operation k in job j , $j = 1, \dots, N$, $i = 1, \dots, N_j$
s_{ji}	Start time of operation k in job j $j = 1, \dots, N$, $i = 1, \dots, N_j$
μ_m	Set of all operations processed on machine m , $m = 1, \dots, M$
	$(j, i) \in \mu_m$ if operation i in job j requires machine m

Appendix 2

Proofs

Proof of Remark 1

In an optimal solution, if $\sum_{j \in \mathcal{J}} T_j > 0$, then

$$i) \sum_{j \in \mathcal{J}} T_j = \sum_{j \in \mathcal{J}_1} T_j,$$

$$ii) T_j < c(\mathcal{J}_1 \setminus j) + p_j - d_j, \forall j \in \mathcal{J}_1, \text{ and}$$

$$iii) \mathcal{J}_1 \text{ starts at time zero.}$$

PROOF: *i)* Clearly, $\sum_{j \in \mathcal{J}} T_j \geq \sum_{j \in \mathcal{J}_1} T_j$. Hence, if the condition does not hold, there is at least one $j \in \mathcal{J}_2$ such that $T_j > 0$. Let u be the job with the earliest start time in \mathcal{J}_2 . Left shifting all j and all jobs that start before j such that u starts at the completion time of \mathcal{J}_1 will not increase the completion time of these jobs, and because tardiness is a regular measure, will not increase their tardiness as well. Clearly, jobs that start after j is completed will remain unaffected. Repeating this argument for all jobs with positive tardiness provides the required result.

ii) Let this condition not hold for some $j \in \mathcal{J}$. Then modifying the schedule such that j starts at time $c(\mathcal{J}_1 \setminus j)$ will not increase its tardiness, while it will increase its release time. Repeating this argument as often as required gives the desired result.

iii) Follows in a straightforward manner from the observation that the tardiness of any job cannot increase if it is started earlier.

Proof of Remark 2

In each block B_k , $k = 1, \dots, K$ in an optimal schedule, there exists one critical job j such that

$$c_j = d_j.$$

PROOF: Because each job j in \mathcal{J}_2 is early, $c_j \leq d_j$. Suppose the condition stated in the remark is not satisfied in block B_k . Let $\tau_j = d_j - c_j$, and

$$u = \arg \min_{j \in B_k} \{\tau_j\}$$

Then deferring the start time of each job in B_k by τ_u increases job release times without delaying any job. Such a right shift will result in u being completed exactly at its due date, and it will, therefore, be the critical job.

Appendix 3

Algorithm for Solving BMRP

1. *Initialization:*

- a) Renumber all jobs as per EDD.
- b) Set $\mathcal{J}_1 = \emptyset$, $\mathcal{J}_2 = \mathcal{J}$.
- c) Determine the maximal partition of \mathcal{J}_2 and the resulting set of job blocks $\{B_k\}$, $k = 1, \dots, K$.

2. *Sequencing Jobs within Blocks:*

- a) For each block $B_k \in \mathcal{J}_2$, solve BMRP. Set

$$D_k = \min_{j \in B_k} \{d_j | c_j = C_k\}; \quad P_k = \max_{j \in B_k} \{c_j\} - \min_{j \in B_k} \{s_j\}$$

If $D_k \geq P_k$, go to Step 3. Otherwise, go to Step 2b.

- b) Set $\mathcal{J}_1 = \mathcal{J}_1 \cup B_k$. Solve ICP1 for \mathcal{J}_1 . Update P_k based on the revised schedule. If $D_k < P_k$, then set $\mathcal{J}_2 = \mathcal{J}_2 \setminus B_k$.

3. *Block Sequencing:* If $\mathcal{J}_1 \neq \emptyset$, set $R_1 = 0$. Starting with the first block in \mathcal{J}_2 , schedule each block $B_k \in \mathcal{J}_2$ such that $C_k = D_k$. If $S_k = C_k - P_k < C_{k-1}$ for any block k , go to Step 4. Otherwise, stop.

4. *Block Merging:* Set $K = K - 1$. If $k = 1$, merge B_1 with \mathcal{J}_1 , reschedule jobs within \mathcal{J}_1 by solving ICP1. Else, merge B_k with B_{k-1} . Solve BMRP for the block thus formed. Update P_{k-1} and D_{k-1} accordingly. Go to Step 3.

Algorithm for Obtaining the Maximal Partition

In the following algorithm, it is assumed that all jobs are numbered in the EDD order. Also, J denotes the set of jobs already assigned to blocks, and $J^c = \mathcal{J} \setminus J$. The algorithm steps are given below.

1. Set $k = 1$; $J = \emptyset$; $J^c = \mathcal{J}$; $j_1 = 1$. Compute $d_j - p_j$, $j = 1, \dots, N$.
2. Find the largest job index j_2 in J^c such that

$$d_{j_2} - p_{j_2} < d_{j_1}.$$

Assign jobs $\{j_1, \dots, j_2\}$ to block k .

3. Set $J^c = J^c \setminus \{j_1, \dots, j_2\}$. If $J^c = \emptyset$, set $K = k$ and stop. Else, set $k \leftarrow k + 1$, $j_1 \leftarrow j_2 + 1$. Go to Step 2.

REFERENCES

1. Ahmadi, R. H. and U. Bagchi (1987), "Just-in-Time Scheduling in Deadline Constrained Environments," Working Paper, University of Texas, Austin, TX.
2. Baker, K. R. (1984a), "The Effects of Input Control in a Simple Scheduling Model," *Journal of Operations Management*, Vol. 4, 99-112.
3. Baker K. R. (1984b), "Sequencing Rules and Due date Assignments in a Job Shop," *Management Science*, Vol. 30, 1093-1104.
4. Baker, K. R. and G. D. Scudder (1990), "Sequencing with Earliness and Tardiness Penalties: A Review," *Operations Research*, Vol. 38, 22-36.
5. Carlier, J. (1982), "The One-Machine Scheduling Problem," *European Journal of Operational Research*, Vol. 11, 42-47.
6. Chand, S. and H. Schneeberger (1985), "A Two-Stage Approximation Method for the Single Machine Weighted Earliness Problem," Working Paper # 424, Graduate School of Business Administration, The University of Michigan, Ann Arbor, MI.
7. Chand, S. and H. Schneeberger (1986), "A Note on the Single Machine Scheduling Problem with Minimum Weighted Completion Time and Maximum Allowable Tardiness," *Naval Research Logistics Quarterly*, Vol. 33, 551-557.
8. Hariri, A. M. A. and C. N. Potts (1983), "An Algorithm for Single Machine Sequencing with Release Dates to Minimize To Weighted Completion Time," *Discrete Applied Mathematics*, Vol. 5, 99-109.
9. Irastorza, J. C. and R. H. Deanne (1974), "A Loading and Balancing Methodology for Job Shop Control," *AIIE Transactions*, Vol. 6, 302-307.
10. McMahon, G. and M. Florian (1975), "On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness," *Operations Research*, Vol. 23, 475-482.
11. Raman, N., F. B. Talbot and R. V. Rachamadugu (1989), "Scheduling a General Flexible Manufacturing System to Minimize Tardiness Related Costs," BEBR

Faculty Working Paper # 89-1547, Univ. of Illinois at Urbana-Champaign, Champaign, IL.

12. Rinnooy Kan, A. H. G. (1976), *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague, Netherlands.
13. Shimoyashiro, S., K. Isoda and H. Awane (1984), "Input Scheduling and Load Balance Control for a Job Shop," *International Journal of Production Research*, Vol. 22, 597-605.
14. Smith, W. E. (1956), "Various Optimizers for Single Stage Production," *Naval Research Logistics Quarterly*, Vol. 3, 56-66.
15. Stecke, K. E. (1983), "Formulation and Solution of Nonlinear Integer Programming Problems for Flexible Manufacturing Systems," *Management Science*, Vol. 29, 273-288.
16. Vepsalainen, A. P. J. and T. E. Morton (1987), "Priority Rules for Job Shops with Weighted Tardiness Costs," *Management Science*, Vol. 33, 1035-1047.

TABLE 1

Comparison of ICP and MOD – Values of (z_1, z_2)
Tight Due Dates

R	F	$(z_1; z_2)$		$z_2(ICP)/z_2(MOD)$	CPU
		MOD	ICP		
0.5	0.05	1.76; 0.54	1.72; 0.71	1.31	2.442
	0.10	0.63; 0.34	0.62; 0.42	1.24	1.785
	0.15	0.12; 0.26	0.12; 0.34	1.31	2.002
	0.20	0.00; 0.22	0.00; 0.35	1.59	3.804
1.5	0.05	1.52; 0.54	1.49; 0.70	1.30	1.757
	0.10	0.51; 0.39	0.50; 0.46	1.18	1.851
	0.15	0.08; 0.31	0.08; 0.39	1.26	2.217
	0.20	0.02; 0.24	0.02; 0.62	2.58	4.952

TABLE 2

Comparison of ICP and MOD – Values of (z_1, z_2)
Loose Due Dates

R	F	$(z_1; z_2)$		$z_2(ICP)/z_2(MOD)$	CPU
		MOD	ICP		
0.5	0.30	0.00; 0.16	0.00; 0.95	5.94	1.889
	0.40	0.00; 0.12	0.00; 0.77	6.42	1.831
	0.50	0.00; 0.10	0.00; 0.83	8.30	1.654
	0.60	0.00; 0.08	0.00; 0.88	11.00	1.204
1.5	0.30	0.00; 0.12	0.00; 0.74	6.17	1.359
	0.40	0.00; 0.12	0.00; 0.82	6.83	0.912
	0.50	0.00; 0.10	0.00; 0.88	8.80	1.758
	0.60	0.00; 0.08	0.00; 0.91	11.34	0.066

TABLE 3

Values of $z_2(ICP)/UB$ for a Two Machine Flow Shop

NJ	F	<i>Minimum</i>	<i>Average</i>	Maximum
25	0.6	0.952	0.973	0.996
	0.7	0.963	0.979	0.997
	0.8	0.970	0.983	0.997
	0.9	0.975	0.986	0.998
	1.0	0.978	0.988	0.998
50	0.6	0.956	0.972	0.992
	0.7	0.965	0.978	0.994
	0.8	0.972	0.982	0.995
	0.9	0.976	0.985	0.996
	1.0	0.979	0.987	0.997

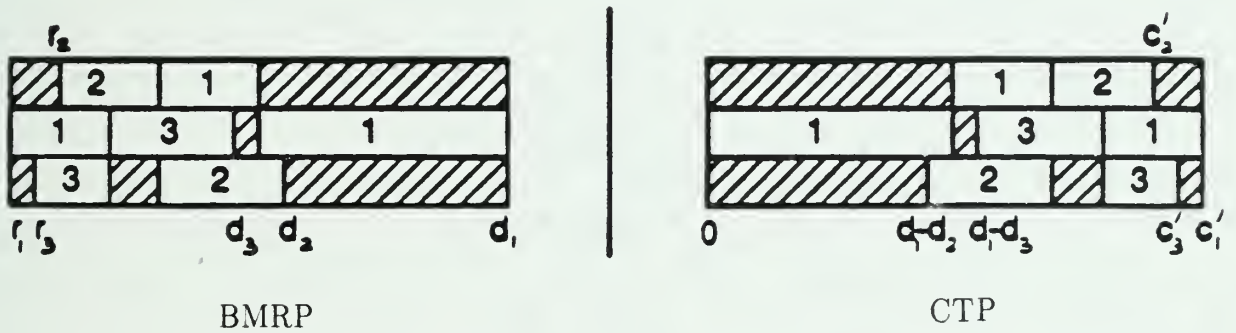


Figure 1 – Equivalence of BMRP and CTP

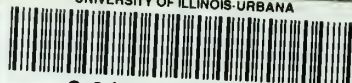
HECKMAN
BINDERY INC.



JUN 95

Bound-To-Please® N. MANCHESTER,
INDIANA 46962

UNIVERSITY OF ILLINOIS-URBANA



3 0112 060295901